Software Licenses and Dependencies

17-313 Fall 2024

Foundations of Software Engineering

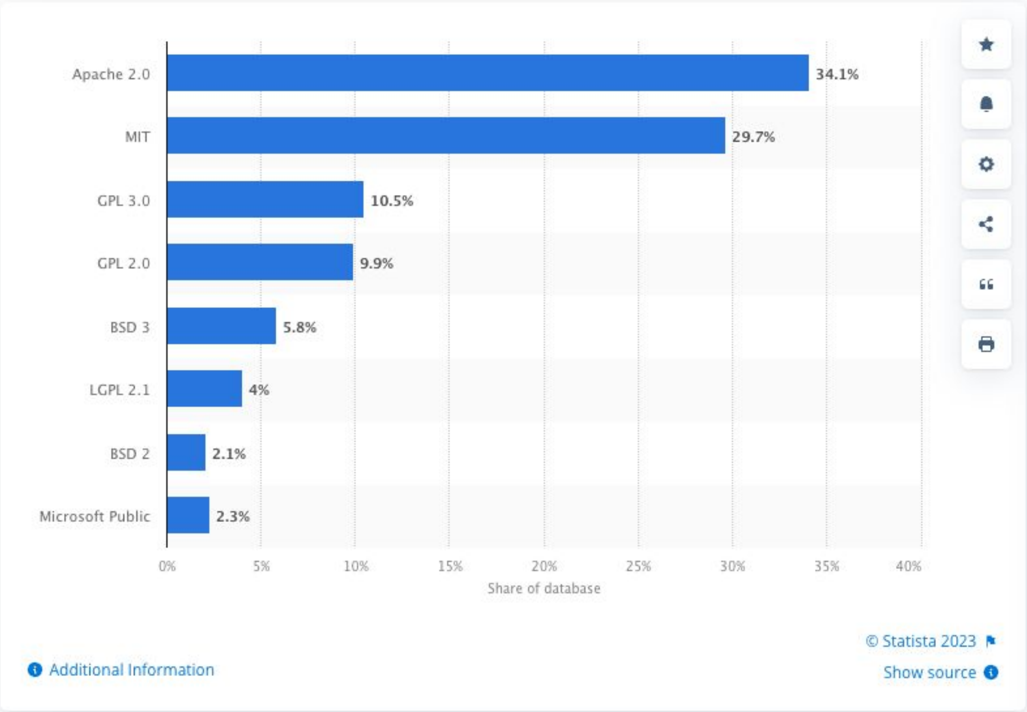
https://cmu-17313q.github.io

Eduardo Feo Flushing

Why learn about licenses?

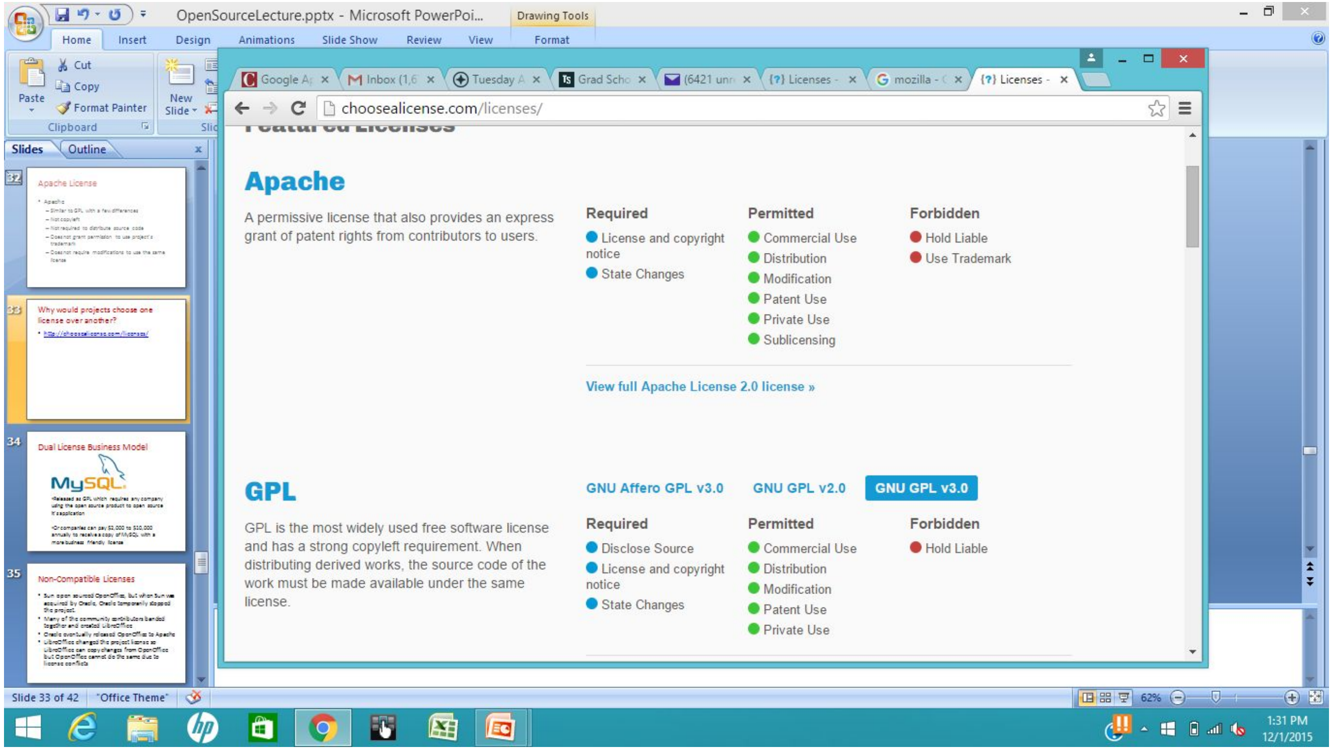
- Companies will avoid certain licenses – commonly the copyleft licenses
- Specific licenses may provide competitive advantages
- You may eventually want to release open source software or become more involved in an open source project

Most popular open source licenses worldwide in 2021



https://www.statista.com/statistics/1245643/worldwide-leading-open-source-licenses/

Which license to choose?



choosealicense.com/licenses/

Apache

A permissive license that also provides an express grant of patent rights from contributors to users.

Required	Permitted	Forbidden
● License and copyright notice	● Commercial Use	● Hold Liable
● State Changes	● Distribution	● Use Trademark
	● Modification	
	● Patent Use	
	● Private Use	
	● Sublicensing	

View full Apache License 2.0 license »

GPL

GNU Affero GPL v3.0 GNU GPL v2.0 GNU GPL v3.0

GPL is the most widely used free software license and has a strong copyleft requirement. When distributing derived works, the source code of the work must be made available under the same license.

Required	Permitted	Forbidden
● Disclose Source	● Commercial Use	● Hold Liable
● License and copyright notice	● Distribution	
● State Changes	● Modification	
	● Patent Use	
	● Private Use	

OpenSourceLecture.pptx - Microsoft PowerPoi...
Drawing Tools
Home
Insert
Design
Animations
Slide Show
Review
View
Format
Paste
Cut
Copy
Format Painter
Clipboard
New Slide
Slides
Outline
32
Apache License
Apache
– Similar to GPL with a few differences
– Not copyleft
– Not required to distribute source code
– Does not grant permission to use project's trademark
– Does not require modifications to use the same license
33
Why would projects choose one license over another?
http://choosealicense.com/licenses/
34
Dual License Business Model
MySQL
Released as GPL which requires any company using the open source product to open source it's application
Or companies can pay $2,000 to $10,000 annually to receive a copy of MySQL with a more business friendly license
35
Non-Compatible Licenses
Sun open sourced OpenOffice, but when Sun was acquired by Oracle, Oracle temporarily stopped the project.
Many of the community contributors banded together and created LibreOffice
Oracle eventually released OpenOffice to Apache
LibreOffice changed the project license so LibreOffice can copy changes from OpenOffice but OpenOffice cannot do the same due to license conflicts
Slide 33 of 42
"Office Theme"
62%
Google Ap
Inbox (1,6
Tuesday A
Grad Scho
(6421 unr
{?} Licenses -
mozilla -
{?} Licenses -
1:31 PM
12/1/2015

Activity: Choosing the Appropriate License

- Analyze a given software scenario and select the most appropriate license from the following: GPL, LGPL, MIT, BSD, Apache.
- In groups of 3-4, read the provided descriptions of a software project and its goals (e.g., commercial use, community contributions, patent protections).
- Consider the Key Factors: Discuss with your group the following aspects:
  - Does the project need copyleft protections or permissive terms?
  - Are there concerns about proprietary use or redistribution?
  - Is patent protection important?
- Choose the license that best aligns with the project's requirements and justify your decision.

GNU General Public License: The **Copyleft** License

- Nobody should be restricted by the software they use. There are four freedoms that every user should have:
  - the freedom to use the software for any purpose,
  - the freedom to change the software to suit your needs,
  - the freedom to share the software with your friends and neighbors, and
  - the freedom to share the changes you make.
- Code must be made available
- Any modifications must be re-licensed **under the same license (copyleft)**

GPL 2.0 and 3.0 – Addresses free software problems

- **GPLv2**
  - Court ruling cannot nullify the license and if a court decision and this license contradict in distribution requirements, then the software cannot be distributed
  - Implicitly grants users a license to any patents held by contributors that are required to use the software.
- **GPLv3**
  - Clarifications on patent grants and prevent “Tivoization”
  - TiVo used Linux, licensed under GPLv2, as the operating system for its DVR devices.
  - While TiVo complied with GPLv2 by releasing the source code of the Linux modifications they made, they implemented hardware restrictions:
  - The device was designed to run only the software signed with TiVo's cryptographic keys.
  - This meant users could view and modify the source code but were unable to run their modified versions on the TiVo hardware.

- Compatibility issues

MIT License

- Must retain copyright credit
- Software is provided as is
- Authors are not liable for software
- Unlike the GPL, it does not require that derivative works be open source
- No other restrictions

LGPL

- Software must be a library
- Similar to GPL but no copyleft requirement

BSD License

- No liability and provided as is.
- Copyright statement must be included in source and binary
- The copyright holder does not endorse any extensions without explicit written consent

Apache License

- Similar to MIT with a few differences
- Not copyleft
- Not required to distribute source code
- Does not grant permission to use project's trademark
- Does not require modifications to use the same license
- Allows proprietary use
- Explicit grant of patent rights

"I am developing a software tool and want to release it with a very permissive license so others can use, modify, and redistribute it, even in proprietary projects, with minimal obligations."

"I have a proprietary application but want to use an open-source library. I need to ensure that I can link the library without being forced to open-source my application."

"I want to release my software under an open-source license but ensure that users are granted explicit patent rights to avoid legal disputes."

"I want to encourage adoption of my open-source software in both open-source and proprietary ecosystems without imposing strict copyleft requirements but still ensuring credit for my work."

"I am building an open-source library and want to ensure that anyone who modifies and redistributes my code must also open-source their changes."

"I want to release my software under an open-source license that allows maximum adoption with minimal obligations for users, but I also want to ensure my name or organization is not used to promote derived works without permission."

I am building an application that includes both open-source components (e.g., framework, or API) and proprietary modules. I want to ensure I can combine these without legally risking the proprietary aspects of my code."

Dual License Business Model

- Released as GPL which requires a company using the open source product to open source it’s application

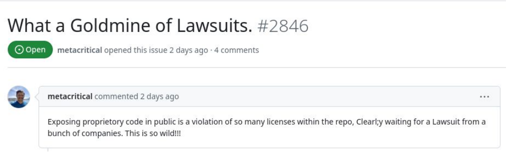
- Or companies can pay $2,000 to $10,000 annually to receive a copy of MySQL with a more business friendly license

Risk: Incompatible Licenses

- Sun open sourced OpenOffice, but when Sun was acquired by Oracle, Oracle temporarily stopped the project.
- Many of the community contributors banded together and created LibreOffice
- Oracle eventually released OpenOffice to Apache
- LibreOffice changed the project license so LibreOffice can copy changes from OpenOffice but OpenOffice cannot do the same due to license conflicts

What not to do

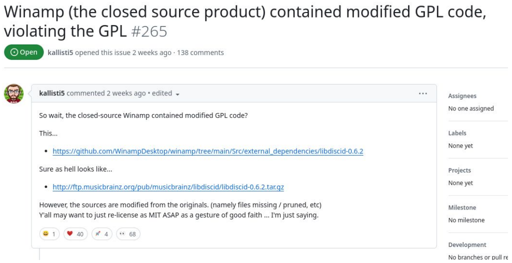
- **Winamp** was a popular media player in the late 1990s and early 2000s
  - Open-source its code to revive community interest and foster further development and innovation.
- Source code release inadvertently included proprietary components from Microsoft, Intel, Dolby, and the SHOUTcast server software
- The initial release under the *"Winamp Collaborative License"* imposed restrictions that **contradicted** open-source principles
  - Prohibiting forks and distribution of modified versions
- Discovery of GPL Code:
  - After releasing the Winamp source code, it was found to include GPL-licensed components.


What a Goldmine of Lawsuits. #2846

Open metacritical opened this issue 2 days ago · 4 comments

metacritical commented 2 days ago

Exposing proprietary code in public is a violation of so many licenses within the repo, Clearl;y waiting for a Lawsuit from a bunch of companies. This is so wild!!!


Winamp (the closed source product) contained modified GPL code, violating the GPL #265

Open kallisti5 opened this issue 2 weeks ago · 138 comments

kallisti5 commented 2 weeks ago • edited

So wait, the closed-source Winamp contained modified GPL code?

This...

- https://github.com/WinampDesktop/winamp/tree/main/Src/external_dependencies/libdiscid-0.6.2

Sure as hell looks like...

- http://ftp.musicbrainz.org/pub/musicbrainz/libdiscid/libdiscid-0.6.2.tar.gz

However, the sources are modified from the originals. (namely files missing / pruned, etc)
Y'all may want to just re-license as MIT ASAP as a gesture of good faith ... I'm just saying.

😄 1 ❤ 40 🚀 4 👀 68

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull re

Producing Open Source

- Like companies, different project have different:
  - Structures
  - Cultures
  - Organizations

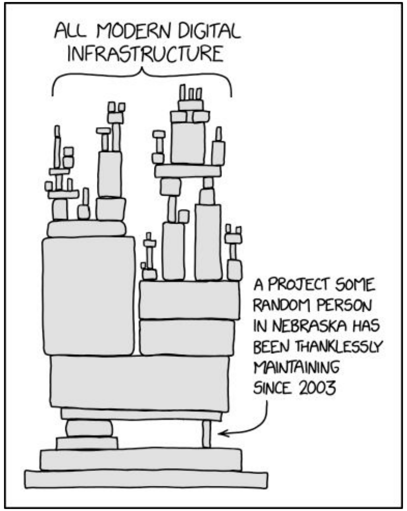
- Many open source projects are supported by a legal entity, such as a foundation.

Producing Open Source: Governance

- Consensus
  - Apache
  - Rust
  - (many others)
- Dictator
  - Python
  - Linux
- Corporate

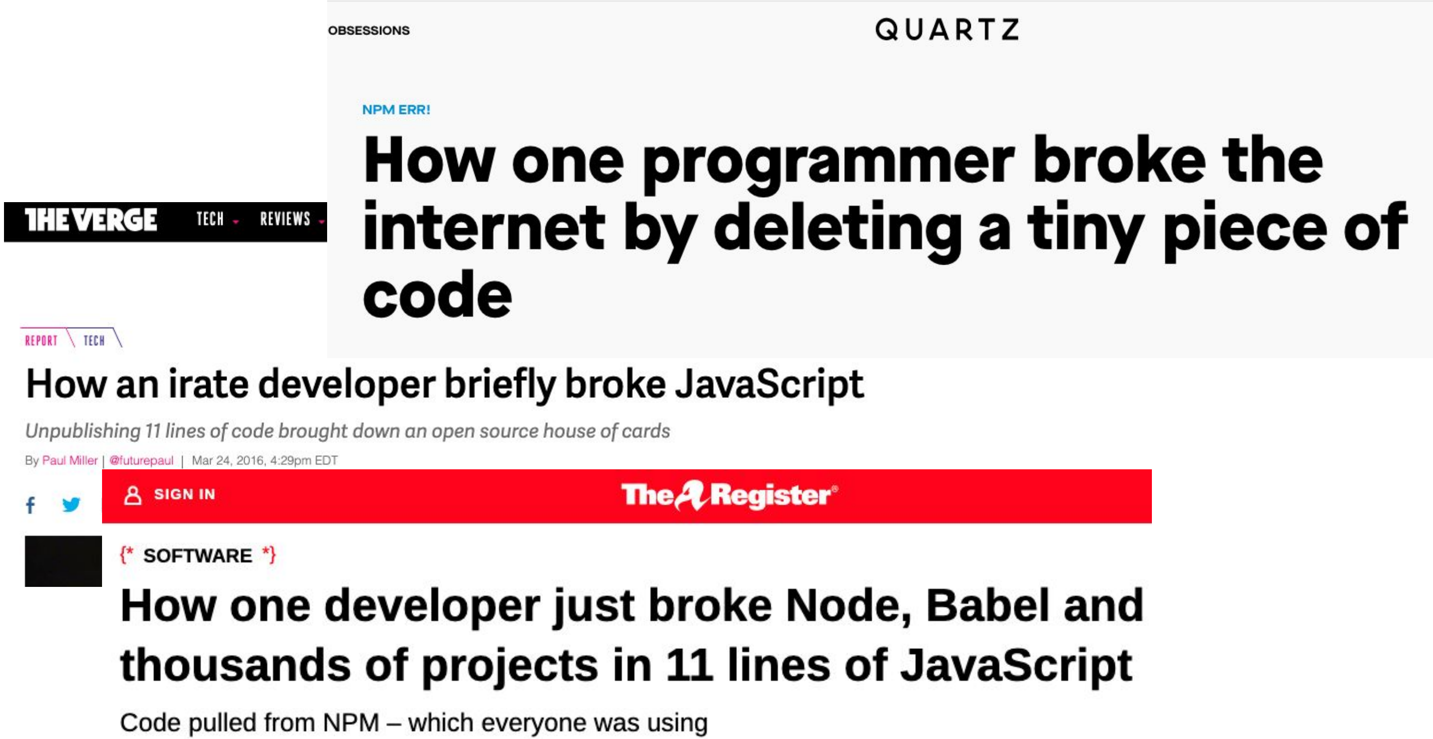
Producing Open Source: Skills

- Written communication
  - Email, chat, and design documents are core to asynchronous work
- “Thick” skin
- Technical ability
- Political ability


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Software Dependencies

Left-pad (March 22, 2016)



OBSESSIONS

QUARTZ

NPM ERR!

How one programmer broke the internet by deleting a tiny piece of code

THE VERGE TECH REVIEWS

REPORT TECH

How an irate developer briefly broke JavaScript

Unpublishing 11 lines of code brought down an open source house of cards

By Paul Miller | @futurepaul | Mar 24, 2016, 4:29pm EDT

SIGN IN

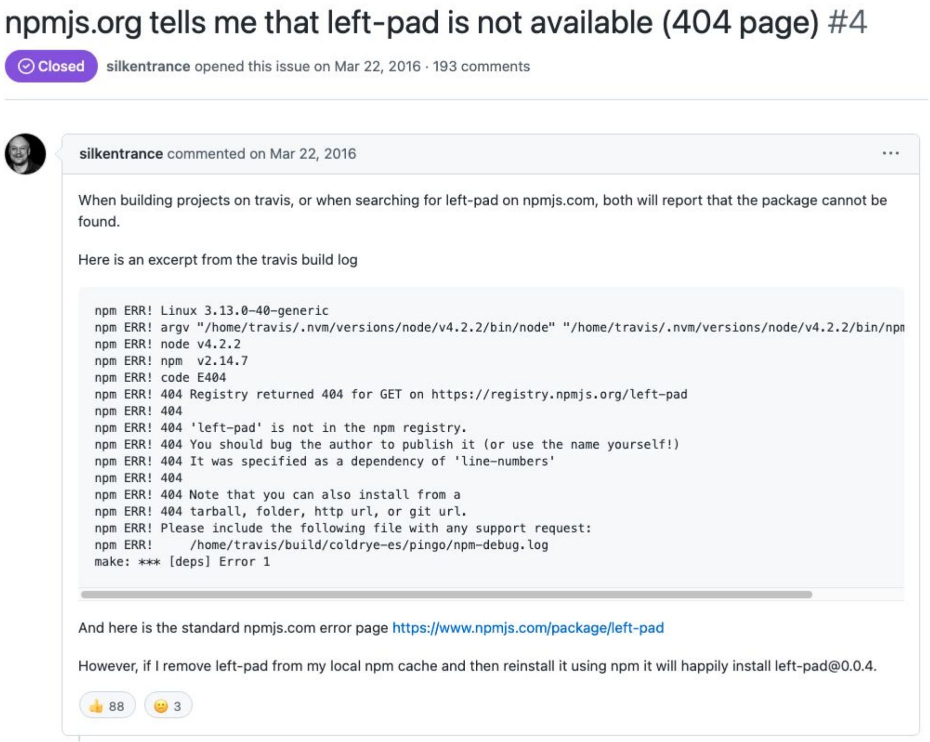
The Register

{* SOFTWARE *}

How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

Left-pad (March 22, 2016)



npmjs.org tells me that left-pad is not available (404 page) #4

Closed **silkentrance** opened this issue on Mar 22, 2016 · 193 comments

silkentrance commented on Mar 22, 2016 ···

When building projects on travis, or when searching for left-pad on npmjs.com, both will report that the package cannot be found.

Here is an excerpt from the travis build log

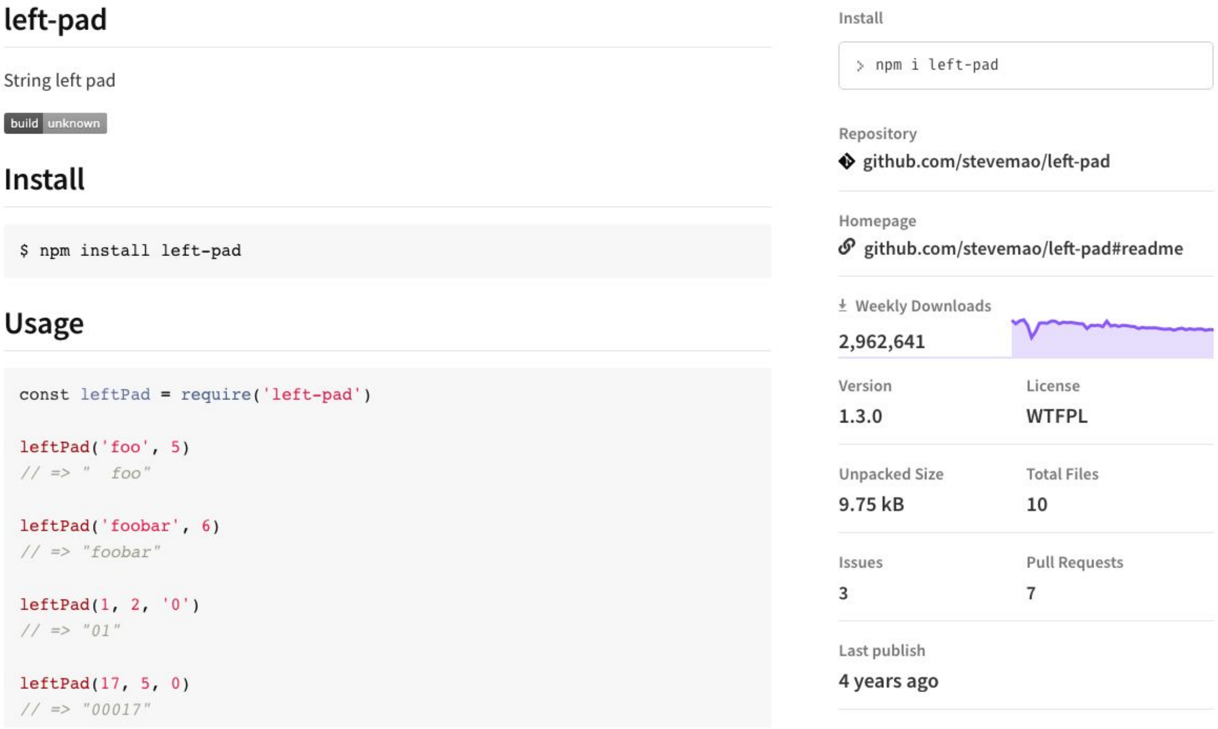
```
npm ERR! Linux 3.13.0-40-generic
npm ERR! argv "/home/travis/.nvm/versions/node/v4.2.2/bin/node" "/home/travis/.nvm/versions/node/v4.2.2/bin/npm
npm ERR! node v4.2.2
npm ERR! npm  v2.14.7
npm ERR! code E404
npm ERR! 404 Registry returned 404 for GET on https://registry.npmjs.org/left-pad
npm ERR! 404
npm ERR! 404 'left-pad' is not in the npm registry.
npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
npm ERR! 404 It was specified as a dependency of 'line-numbers'
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.
npm ERR! Please include the following file with any support request:
npm ERR!     /home/travis/build/coldrye-es/pingo/npm-debug.log
make: *** [deps] Error 1
```

And here is the standard npmjs.com error page https://www.npmjs.com/package/left-pad

However, if I remove left-pad from my local npm cache and then reinstall it using npm it will happily install left-pad@0.0.4.

👍 88 😕 3

Left-pad (Docs)



left-pad

String left pad

build unknown

Install

```
$ npm install left-pad
```

Usage

```
const leftPad = require('left-pad')

leftPad('foo', 5)
// => "  foo"

leftPad('foobar', 6)
// => "foobar"

leftPad(1, 2, '0')
// => "01"

leftPad(17, 5, 0)
// => "00017"
```

Install

```
> npm i left-pad
```

Repository

github.com/stevemao/left-pad

Homepage

github.com/stevemao/left-pad#readme

Weekly Downloads

2,962,641

Version	License
1.3.0	WTFPL

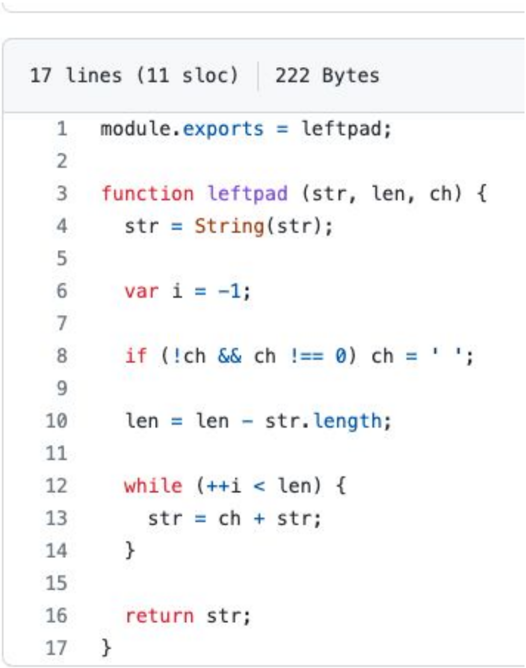
Unpacked Size	Total Files
9.75 kB	10

Issues	Pull Requests
3	7

Last publish

4 years ago

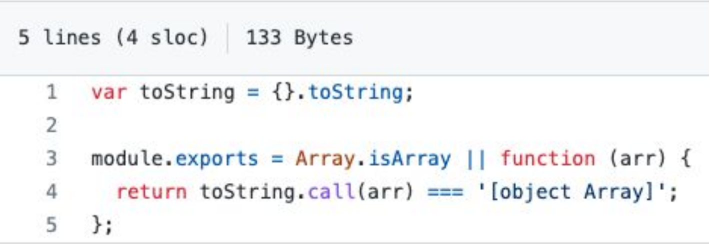
Left-pad (Source Code)



17 lines (11 sloc) | 222 Bytes

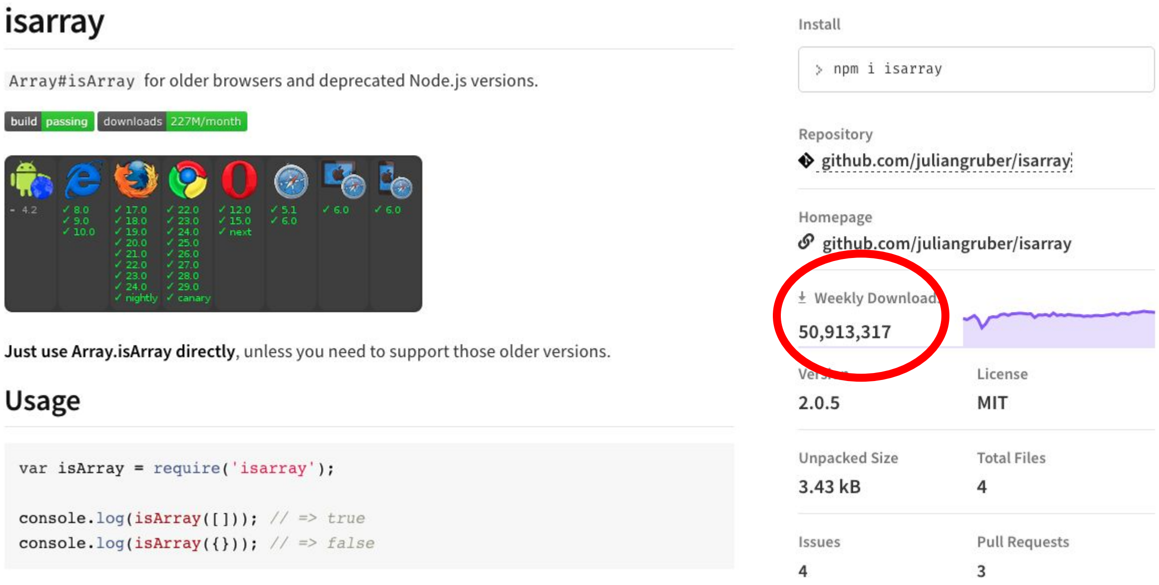
```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

See also: isArray



5 lines (4 sloc) | 133 Bytes

```
var toString = {}.toString;

module.exports = Array.isArray || function (arr) {
  return toString.call(arr) === '[object Array]';
};
```



isarray

`Array#isArray` for older browsers and deprecated Node.js versions.

build passing
downloads 227M/month
- 4.2
✓ 8.0 ✓ 9.0 ✓ 10.0
✓ 17.0 ✓ 18.0 ✓ 19.0 ✓ 20.0 ✓ 21.0 ✓ 22.0 ✓ 23.0 ✓ 24.0 ✓ nightly
✓ 22.0 ✓ 23.0 ✓ 24.0 ✓ 25.0 ✓ 26.0 ✓ 27.0 ✓ 28.0 ✓ 29.0 ✓ canary
✓ 12.0 ✓ 15.0 ✓ next
✓ 5.1 ✓ 6.0
✓ 6.0
✓ 6.0

Just use Array.isArray directly, unless you need to support those older versions.

Usage

```
var isArray = require('isarray');

console.log(isArray([])); // => true
console.log(isArray({})); // => false
```

Install

```
> npm i isarray
```

Repository

github.com/juliangruber/isarray

Homepage

github.com/juliangruber/isarray

Weekly Downloads

50,913,317

Version	License
2.0.5	MIT

Unpacked Size	Total Files
3.43 kB	4

Issues	Pull Requests
4	3

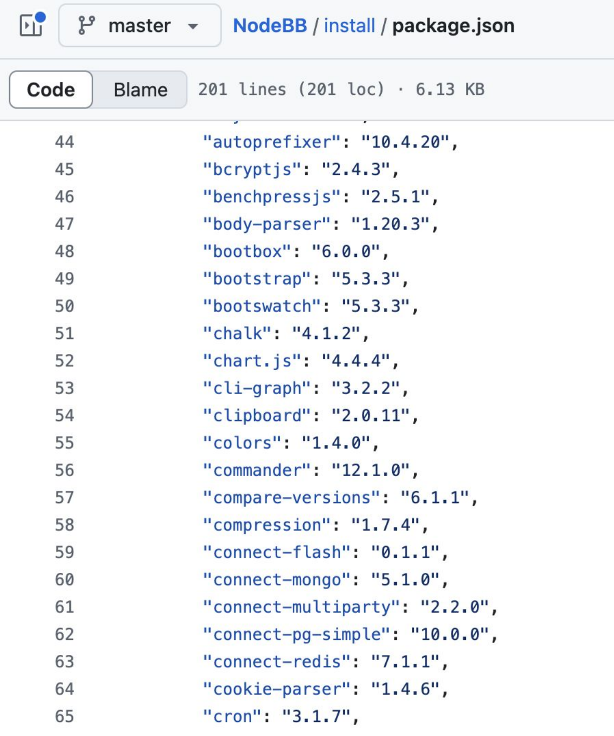
How do software projects manage third-party dependencies on reusable libraries?

- It’s hard
- It’s mostly a mess (everywhere)
- But it’s critical to modern software development

What is a Dependency?

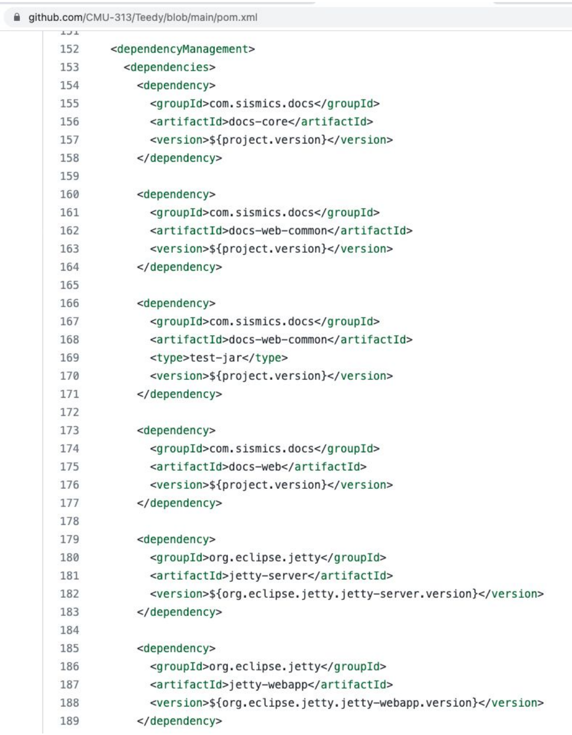
- Core of what most build systems do
  - “Compile” and “Run Tests” is just a fraction of their job
- Examples: Maven, Gradle, NPM, Bazel, …
- **Foo->Bar**: To build Foo, you may need to have a built version of Bar
- Dependency Scopes:
  - **Compile**: Foo uses classes, functions, etc. defined by Bar
  - **Runtime**: Foo uses an abstract API whose implementation is provided by Bar (e.g. logging, database, network or other I/O)
  - **Test**: Foo needs Bar only for tests (e.g. JUnit, mocks)
- Internal vs. External Dependencies
  - Is Bar also built/maintained by your org or is it pulled from elsewhere using a package manager?

Examples of dependency views

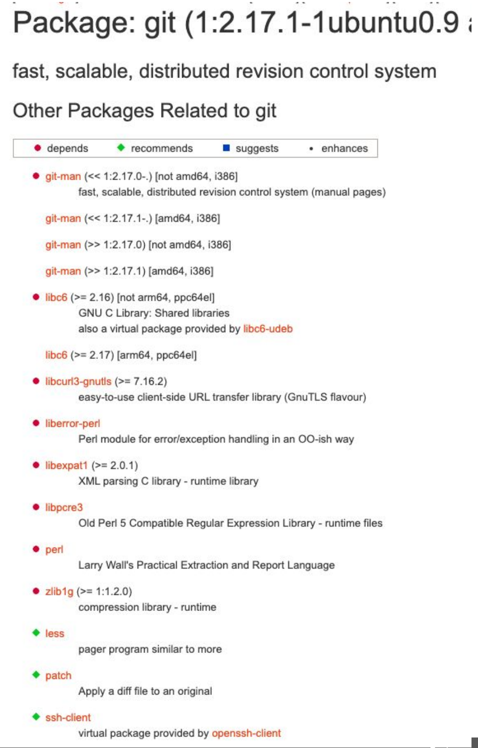


201 lines (201 loc) · 6.13 KB

```
    "autoprefixer": "10.4.20",
    "bcryptjs": "2.4.3",
    "benchpressjs": "2.5.1",
    "body-parser": "1.20.3",
    "bootbox": "6.0.0",
    "bootstrap": "5.3.3",
    "bootswatch": "5.3.3",
    "chalk": "4.1.2",
    "chart.js": "4.4.4",
    "cli-graph": "3.2.2",
    "clipboard": "2.0.11",
    "colors": "1.4.0",
    "commander": "12.1.0",
    "compare-versions": "6.1.1",
    "compression": "1.7.4",
    "connect-flash": "0.1.1",
    "connect-mongo": "5.1.0",
    "connect-multiparty": "2.2.0",
    "connect-pg-simple": "10.0.0",
    "connect-redis": "7.1.1",
    "cookie-parser": "1.4.6",
    "cron": "3.1.7",
```



```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.sismics.docs</groupId>
      <artifactId>docs-core</artifactId>
      <version>${project.version}</version>
    </dependency>

    <dependency>
      <groupId>com.sismics.docs</groupId>
      <artifactId>docs-web-common</artifactId>
      <version>${project.version}</version>
    </dependency>

    <dependency>
      <groupId>com.sismics.docs</groupId>
      <artifactId>docs-web-common</artifactId>
      <type>test-jar</type>
      <version>${project.version}</version>
    </dependency>

    <dependency>
      <groupId>com.sismics.docs</groupId>
      <artifactId>docs-web</artifactId>
      <version>${project.version}</version>
    </dependency>

    <dependency>
      <groupId>org.eclipse.jetty</groupId>
      <artifactId>jetty-server</artifactId>
      <version>${org.eclipse.jetty.jetty-server.version}</version>
    </dependency>

    <dependency>
      <groupId>org.eclipse.jetty</groupId>
      <artifactId>jetty-webapp</artifactId>
      <version>${org.eclipse.jetty.jetty-webapp.version}</version>
    </dependency>
```



Package: git (1:2.17.1-1ubuntu0.9

fast, scalable, distributed revision control system

Other Packages Related to git

● depends ◆ recommends ■ suggests • enhances

- ● git-man (<< 1:2.17.0-.) [not amd64, i386]
  fast, scalable, distributed revision control system (manual pages)

  git-man (<< 1:2.17.1-.) [amd64, i386]

  git-man (>> 1:2.17.0) [not amd64, i386]

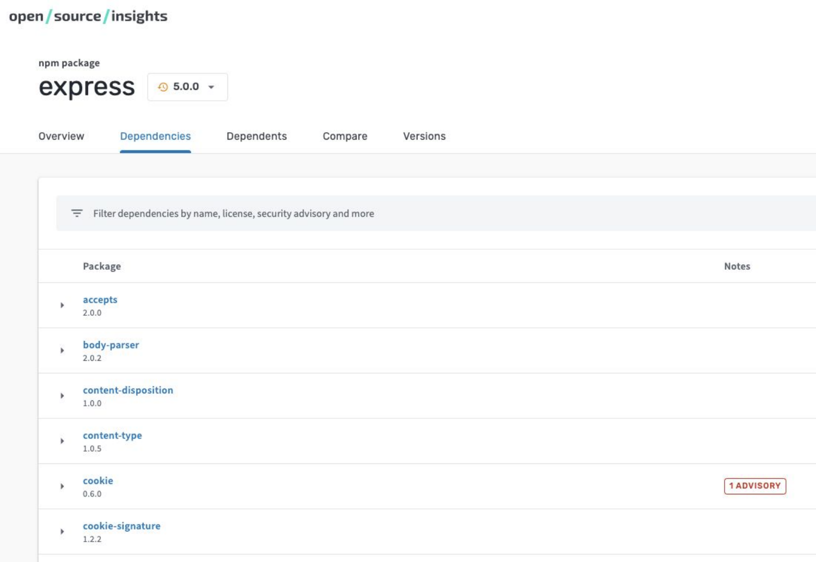
  git-man (>> 1:2.17.1) [amd64, i386]
- ● libc6 (>= 2.16) [not arm64, ppc64el]
  GNU C Library: Shared libraries
  also a virtual package provided by libc6-udeb

  libc6 (>= 2.17) [arm64, ppc64el]
- ● libcurl3-gnutls (>= 7.16.2)
  easy-to-use client-side URL transfer library (GnuTLS flavour)
- ● liberror-perl
  Perl module for error/exception handling in an OO-ish way
- ● libexpat1 (>= 2.0.1)
  XML parsing C library - runtime library
- ● libpcre3
  Old Perl 5 Compatible Regular Expression Library - runtime files
- ● perl
  Larry Wall's Practical Extraction and Report Language
- ● zlib1g (>= 1:1.2.0)
  compression library - runtime
- ◆ less
  pager program similar to more
- ◆ patch
  Apply a diff file to an original
- ◆ ssh-client
  virtual package provided by openssh-client

Where are the dependencies hosted?

- Typically downloaded from dependency servers:
  - Maven Central (https://repo.maven.apache.org/maven2/)
  - Ubuntu Packages for Apt (https://packages.ubuntu.com/)
  - Python Package Index (https://pypi.org/)]
  - NPM Public Registry (https://registry.npmjs.org/)
- Packages need a unique identifier
  - Typically a package name (sometimes owner name) and version
- Custom repositories allowed by most package managers
  - Often used for company-internal packages or cache mirroring
  - Note problems with duplicates (same pkg name in different repositories; some priority order is needed)
- Somebody needs to manage repositories
  - **Availability**: Repository needs to be running
  - **Access Control:** Packages should only be published by owners
  - **Integrity:** Packages should be signed or otherwise verifiable
  - **Uniqueness and archival:** Only one artifact per version
  - **Traceability:** Packages can have metadata pointing to source or tests
  - **Security:** ???

Demo: Deps.dev



open / source / insights

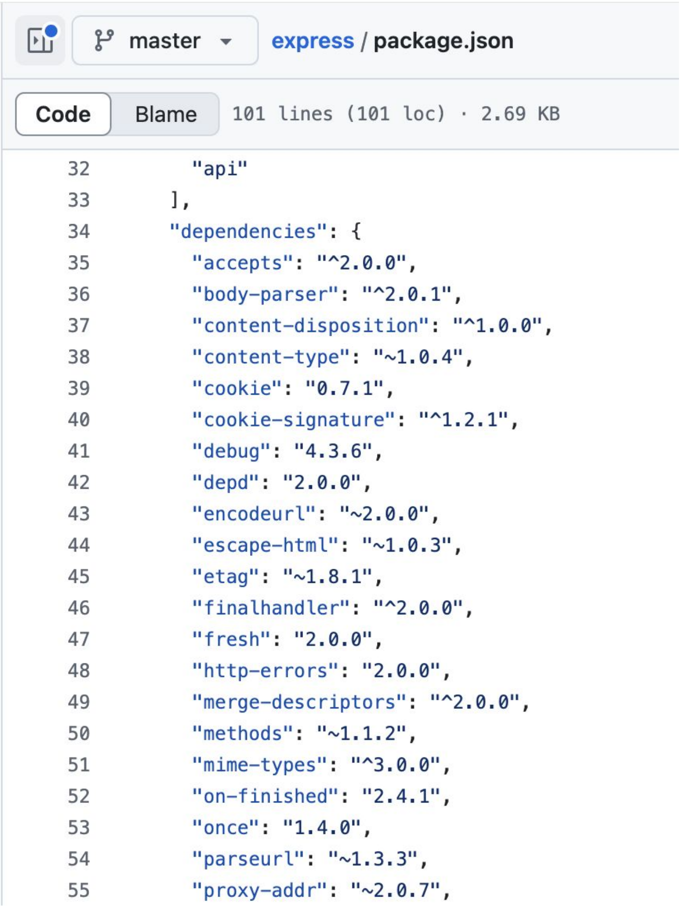
npm package

express 5.0.0

Overview | Dependencies | Dependents | Compare | Versions

Filter dependencies by name, license, security advisory and more

Package	Notes
accepts 2.0.0	
body-parser 2.0.2	
content-disposition 1.0.0	
content-type 1.0.5	
cookie 0.6.0	1 ADVISORY
cookie-signature 1.2.2	



master | express / package.json

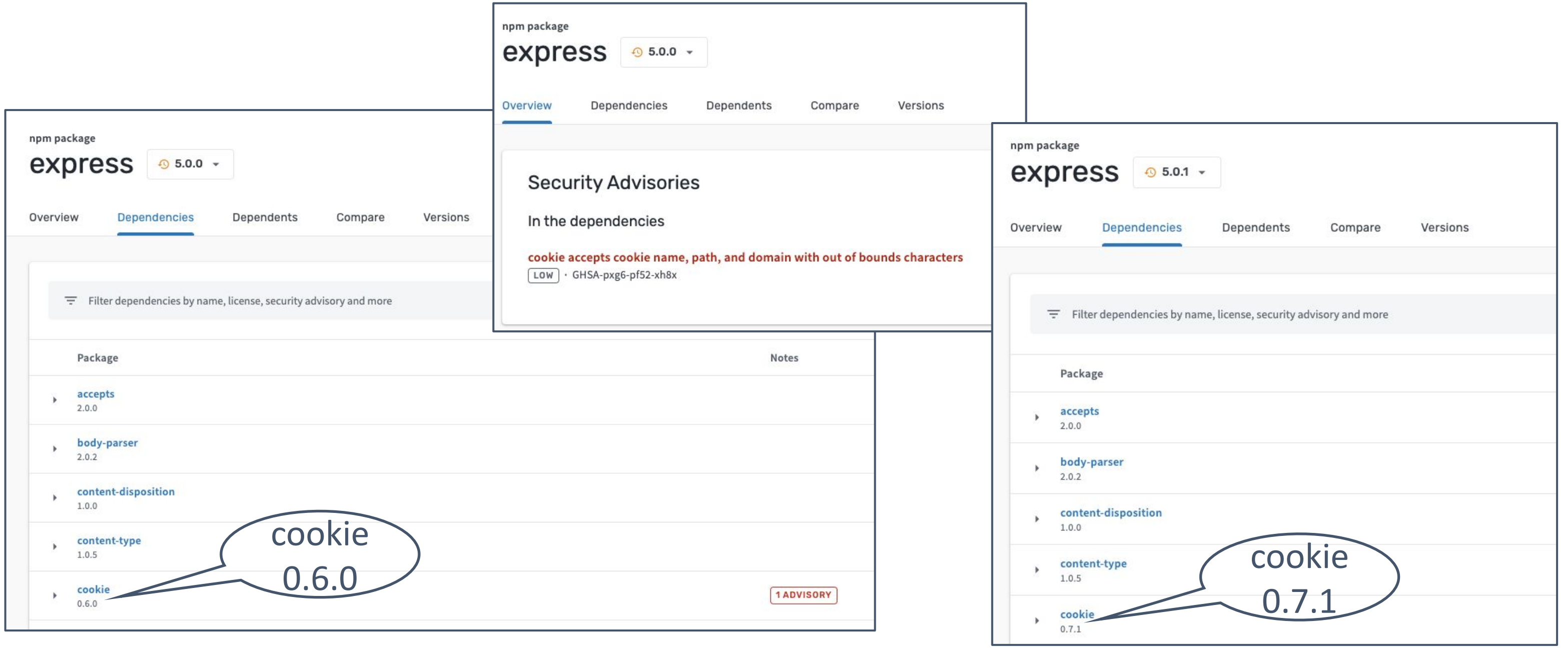
Code | Blame | 101 lines (101 loc) · 2.69 KB

```
    "api"
  ],
  "dependencies": {
    "accepts": "^2.0.0",
    "body-parser": "^2.0.1",
    "content-disposition": "^1.0.0",
    "content-type": "~1.0.4",
    "cookie": "0.7.1",
    "cookie-signature": "^1.2.1",
    "debug": "4.3.6",
    "depd": "2.0.0",
    "encodeurl": "~2.0.0",
    "escape-html": "~1.0.3",
    "etag": "~1.8.1",
    "finalhandler": "^2.0.0",
    "fresh": "2.0.0",
    "http-errors": "2.0.0",
    "merge-descriptors": "^2.0.0",
    "methods": "~1.1.2",
    "mime-types": "^3.0.0",
    "on-finished": "2.4.1",
    "once": "1.4.0",
    "parseurl": "~1.3.3",
    "proxy-addr": "~2.0.7",
```

Dependency Pinning vs. Floating

- Pinning: ”I depend on libFoo 1.5.0”
  - Declares a specific version of the dependency. **Frozen in time.**
- Floating: “I depend on libFoo-latest”
  - Each build will pull the **latest available** libFoo version
  - (Other forms available, e.g. libFoo 1.5.x)

Pinned dependencies requires manual updates in case of security issues



npm package

express 5.0.0

Overview | Dependencies | Dependents | Compare | Versions

Filter dependencies by name, license, security advisory and more

Package	Notes
accepts 2.0.0	
body-parser 2.0.2	
content-disposition 1.0.0	
content-type 1.0.5	
cookie 0.6.0	1 ADVISORY

npm package

express 5.0.0

Overview | Dependencies | Dependents | Compare | Versions

Security Advisories

In the dependencies

cookie accepts cookie name, path, and domain with out of bounds characters

LOW · GHSA-pxg6-pf52-xh8x

npm package

express 5.0.1

Overview | Dependencies | Dependents | Compare | Versions

Filter dependencies by name, license, security advisory and more

Package
accepts 2.0.0
body-parser 2.0.2
content-disposition 1.0.0
content-type 1.0.5
cookie 0.7.1

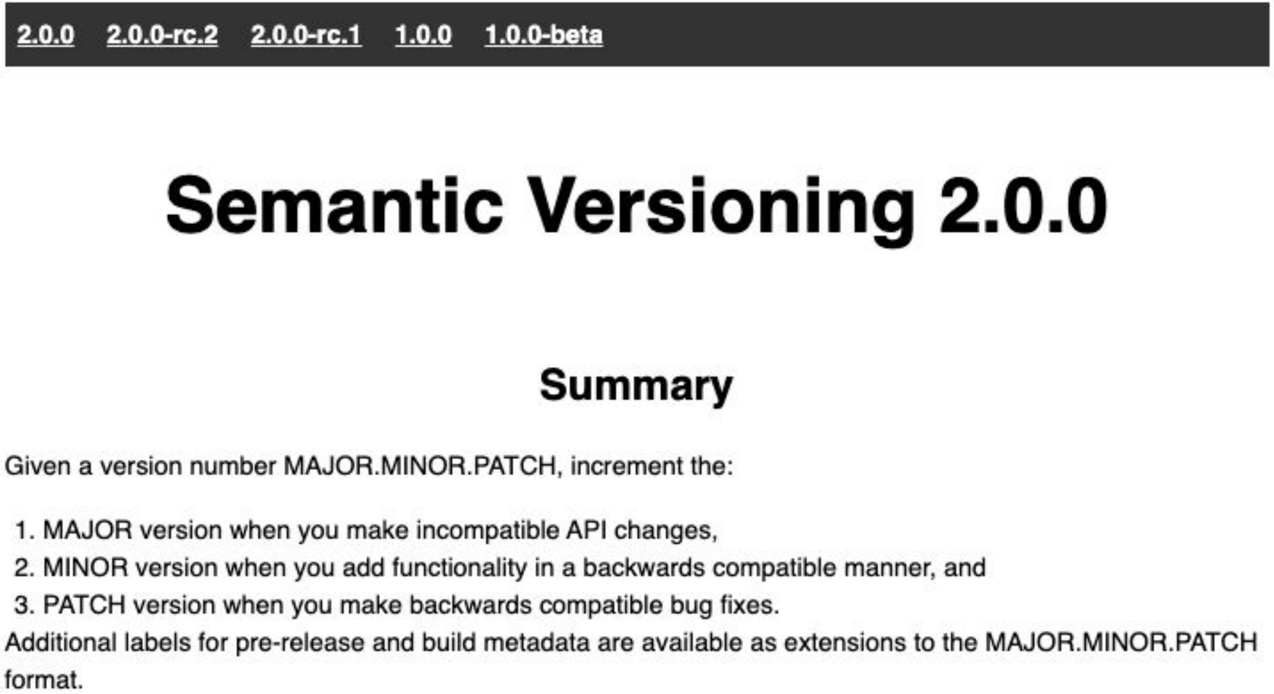
Pinning vs Floating

Pinning Dependencies **(e.g. 1.5.3)**	**Floating Dependencies** **(e.g. 1.x)**
✅ Reproducible builds	❌ Flaky builds (breaking changes)
❌ Can become vulnerable due to dependency bugs	✅ Latest security patches & bug fixes
❌ Have to keep updating dependents as dependencies evolve	✅ Less manual maintenance
✅ Stable network effects	❌ Floats leak transitively (A pin to B floating C; then A still sees changing version of C)

Semantic Versioning

- Widely used convention for versioning releases
  - E.g. 1.2.1, 3.1.0-alpha-1, 3.1.0-alpha-2, 3.1.0-beta-1, 3.1.0-rc1
- Format: {MAJOR} . {MINOR} . {PATCH}
- Each component is ordered (numerically, then lexicographically; release-aware)
  - 1.2.1 < 1.10.1
  - 3.1.0-alpha-1 < 3.1.0-alpha-2 < 3.1.0-beta-1 < 3.1.0-rc1 < 3.1.0
- Contracts:
  - MAJOR updated to indicate breaking changes
    - Same MAJOR version => backward compatibility
  - MINOR updated for additive changes
    - Same MINOR version => API compatibility (important for linking)
  - PATCH updates functionality without new API
    - Ninja edit; usually for bug fixes
- Largely dependent on honor system. No easy way to automatically verify (can you solve it?)

https://semver.org/



Semantic Versioning 2.0.0

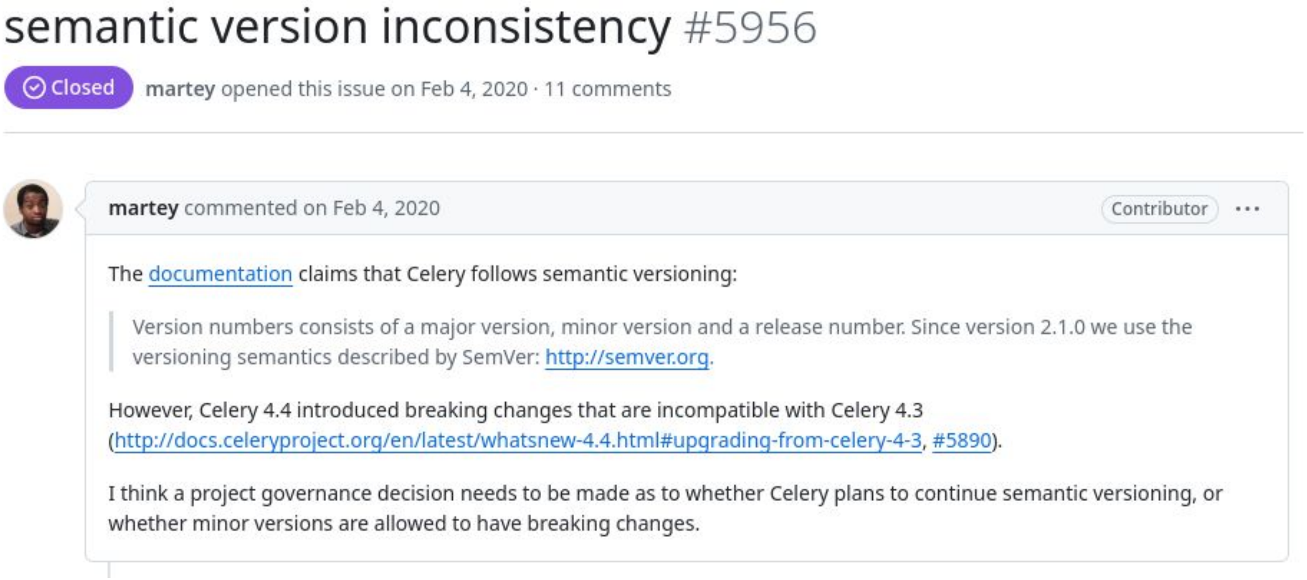
Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

People rely on SemVer contracts



semantic version inconsistency #5956

martey opened this issue on Feb 4, 2020 · 11 comments

martey commented on Feb 4, 2020

The documentation claims that Celery follows semantic versioning:

> Version numbers consists of a major version, minor version and a release number. Since version 2.1.0 we use the versioning semantics described by SemVer: http://semver.org.

However, Celery 4.4 introduced breaking changes that are incompatible with Celery 4.3 (http://docs.celeryproject.org/en/latest/whatsnew-4.4.html#upgrading-from-celery-4-3, #5890).

I think a project governance decision needs to be made as to whether Celery plans to continue semantic versioning, or whether minor versions are allowed to have breaking changes.

See full thread for an “interesting” example of Open Source governance and communication